Machine Learning in the calibration process of Discrete Particle Model: The case with Angle of Repose

Quang Hung Nguyen

Head Supervisor: Thomas Weinhart, Daily Supervisor: Anthony Thornton, Additional member: Chen Kuan.

Multi-Scale Mechanics Group, Faculty of Engineering Technology, University of Twente.

Abstract

This is where, stuff, happens. Literally. But not yet. Please wait.

Contents

1 Introduction **3**

2 Characterisation of granular materials **4**

3 Simulation Method **4**
3.1 Discrete Particle Model 4
3.1.1 Contact Laws 5
3.1.2 Angle of Repose measurement 5
3.2 Material and simulation properties 6

4 Calibration of Discrete Particle Model **7**
4.1 GrainLearning 7
4.1.1 Posterior distribution calculation 7
4.1.2 Effective multi-level sampling 8
4.1.3 Identification of microparameters with GrainLearning 8
4.2 Neural Network 8
4.2.1 Designing a Neural Network 8
4.2.2 Training and evaluation method 9
4.3 Random Forest algorithm 9
4.3.1 Training and evaluation method 10

5 GrainLearning evaluation **10**
5.1 Limestone 10
5.2 Quartz sand 11
5.3 GL discussion 12

6 Supervised model evaluation **13**
6.1 Limestone 13
6.2 Quartz Sand 14
6.3 NN and RF model discussion 16

7 Discussion **16**

8 Conclusion **17**

1 Introduction

Granular material is a family of material characterized by its large bulk of densely packed particles, ranging from nanometers to centimeters [1], and is able to resist deformation and form heaps, i.e., behave like a solid and withstand strong shear force [2]. Simple examples of granular materials include sand, gravel, clays, seeds, nuts, and all ranges of powders such as coffee powder, cement powder, which is shown in figure 1. Furthermore, many processes and equipments in chemical plants use granular materials, such as catalysis, adsorption, and heat exchangers. Granular materials are projected to make about half of the products and three-quarters of the raw materials used in the chemical industry [3]. Thus, understanding how granular materials behave is of great significance.

Figure 1: Examples of Granular Materials [4].

The simulation of granular material's bulk mechanical behavior is done using Discrete Particle Model (DPM, or Discrete Element Method - DEM), which generates the movement of individual particles to capture the macro-scale behavior. The DPM is a family of numerical methods for computing the motion of a large number of particles [5], first proposed by Cundall and Strack in the 1970s [6]. Since the properties of granular materials differ wildly, these simulations require an extensive calibration process designed individually for each type of granular material. Some parameters of the granular material model can be measured directly, such as size distribution or density. However, other parameters are effective parameters (i.e., they result from a simplified particle model) and thus cannot be directly measured. These parameters are then calibrated by choosing a few standard calibration setups (rotating drum, heap test, ring shear cell) and simulating these setups in a DPM simulation, and the missing parameters are determined such that the response of the experimental and simulation setups match.

Recently, coupled with the raise of Machine Learning in other fileds, it has also been applied to solve the calibration problem. This has been done using a Neural Network [7, 8, 9, 10], Genetic Algorithm [11], and a recursive Bayesian sequential Monte-Carlo filtering algorithm named GrainLearning [12]. In this Assignment, three Machine Learning algorithms will be discussed: Neural Network, Random Forest (supervised model), and GrainLearning (unsupervised model). These three algorithms set to treat the calibration problem in two different ways, and likewise, solve it in two different ways: While GrainLearning looks to identify the microparameters from the experimental and DEM simulations's bulk parameters (inverse problem), the supervised models will help generating a database that can map different microparameters combinations to their corresponding bulk parameters, generated by DEM simulations. In other word, NN and RF will learn the built-in relationship between the micro- and macroparameters of the Discrete Particle Model, thus allow a much faster prediction compare to a full DEM simulation. One advantage of GrainLearning compares to other Machine Learning algorithms such as Neural Network is that it is an unsupervised learning algorithm, i.e., it can starts calibrating

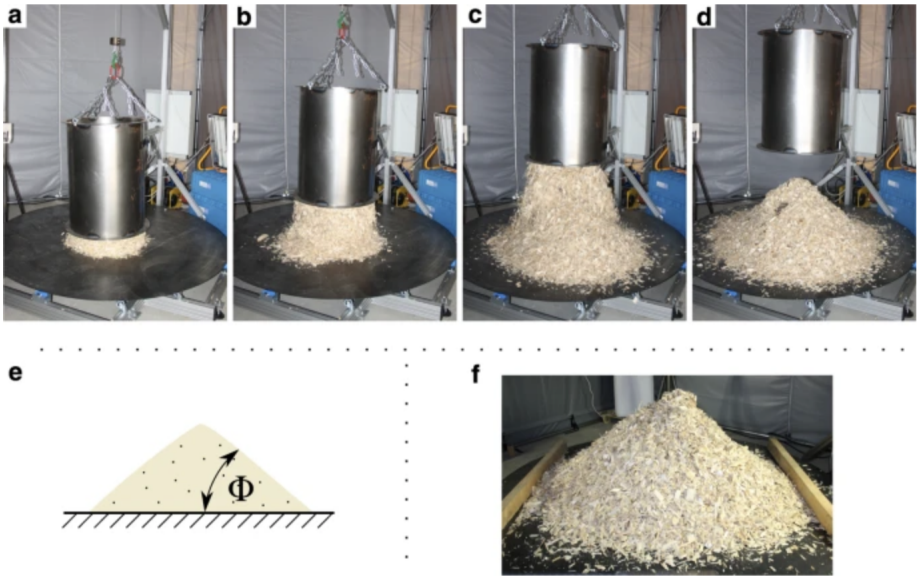

Figure 2: Static Angle of Repose measurement steps [14].

with a minimal amount of input information. However, each material needs calibration for multiple bulk parameters, i.e., Static Angle of Repose, Dynamic Angle of Repose, shear tester, etc. since a set of microparameters that is valid for one bulk parameter might not valid for another. Therefore, scaling up with a Neural Network model might be more simple, since a Neural Network can produces multiple valid combinations for each bulk parameter.

In the next section, the characterisation and experimental method will be discussed, including static Angle of Repose, Discrete Particle Model, contact laws, and the material used in the simulation. Section 4 will discuss in detail different approach and method of each model. The result of GrainLearning will be discuss in section 5, while section 6 discuss the supervised model's performance. Subsequently, section 7 will compare the models, discuss the strength and weakness of each model, and the limitations.

2 Characterisation of granular materials

There are no established standard of characterisation measurements for granular materials. Common measurements include heap test, rotating drum test, linear/ring shear cell test, and the silo flow test,..., in which the output is the bulk parameter, which defines how the granular material behaves in large quantity - such as angle of repose (AoR), shear stress, flow rate, etc.

This research is focused on one of the most important bulk parameters to describe the characteristics of granular materials - the static angle of repose. Static AoR, which is described in Fig. 2, is defined as the angle that granular solids forms when it piled with a flat surface, and is essential to characterise the coarseness and smoothness of materials. This in turn can help designing a process involved with the material - lower static AoR implies more flowable and thus easier to transport with less energy [13].

3 Simulation Method

3.1 Discrete Particle Model

Discrete Particle Model simulates particle motion by applying forces and torques which derives from particle-particle interactions and external influences, on the basis of the given contact law. It performs calculations of kinematics that a given particle i exerts on other particle j, for each particle in the

system, among with the peripheral factors such as gravity and walls. To achieve this results, the particles are assumes to be (1) undeformable - deform therefore implemented as overlap, (2) unbreakable, (3) all internal interactions are due to particle-particle interaction, (4) Each particle pair i, j has only on/viewer.htmle contact point c_{ij} which the forces and torques act on, and (5) all external forces and torques are either body forces and torques or by interacting with a wall [15].

3.1.1 Contact Laws

For each particle i on the system, Eq. 1 describes the internal and external forces, and Eq. 2 describes the torque acting on it [15]:

$$F_i = \sum_{j=1}^{n_p} F_{ij} + \sum_{k=1}^{n_w} F_{ik}^w + F_i^b \tag{1}$$

$$\tau_i = \sum_{j=1}^{n_p} r_{ij} F_{ij} + \tau_{ij} + \sum_{k=1}^{n_w} r_{ik} F_{ik}^w + \tau_{ik}^w + \tau_i^b \tag{2}$$

With F_{ij} interparticle forces, F_{ik}^w the interaction force between each wall and the particle, n_p = number of particles, n_w = number of walls, F_i^b body forces i.e., gravity, and r_{ij} as the branch vector, which connects the particle position r_i with the contact point c_{ij} . The same hold for torques equation, with τ as torque.

The contact law used in the simulations is the Linear Spring-Dashpot model, which is implemented in MercuryDPM as `LinearViscoelasticFrictionReversibleAdhesiveSpecies`. It defines the interaction between two particles i and j as a damped harmonic oscillators [16]:

$$F_{ij}^n = \begin{cases} k_n \delta_{ij}^n + \gamma_n v_{ij}^n & \text{if } \delta_{ij}^n > 0, \\ 0 & \text{else,} \end{cases} \tag{3}$$

In this equation, $k_n > 0$ represents spring stiffness, $\gamma_n \geq 0$ represents the damping coefficient, v_n the normal vector, and δ_{ij}^n is the overlap between the particles. Two particles interact with each other if and only if they overlap. This contact model is simple, has an analytic solution and less computationally expensive [17], while also suitable for large particles [15].

3.1.2 Angle of Repose measurement

In MercuryDPM, Static AoR is measured by a hollow cylinder simulation, which consists of two cylinders (instead of one cylinder and one plate in Fig. 2), with the cylinder's diameter. For simplicity, all particles in the simulation are assumed to be a perfect sphere. After all the particles are poured into the cylinder, it is let to rest until the bulk's kinetic energy is less than 1% compared to the potential energy (steady-state condition). At this point, the top cylinder is removed, and all the particles that have fallen out of the bottom cylinder below $z = 0$ are deleted. This will result in a cone-shaped heap of particles and a drastic increase in kinetic energy due to gravity. The heap will be rested again until it reach a steady-state condition as mentioned above, and then Static AoR can be measured. The measurement will be done twice, since the kinetic energy of the system can be increased again after the first measurement. Figure 3 demonstrates an example simulation on MercuryDPM.

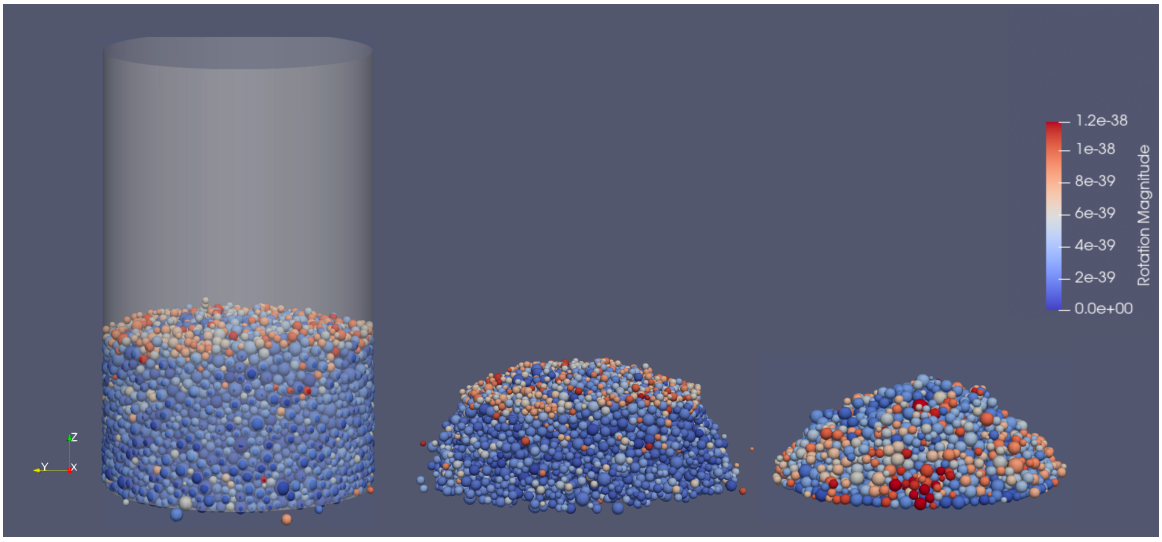

Figure 3: Angle of Repose simulation on MercuryDPM. From left to right: Initial fill stage, wall removed, and final stage.

3.2 Material and simulation properties

Experimental data on quartz sand is provided by Derakhshani et al. [18]. The density of quartz sand is $\rho = 2653\ kg/m^3$, and the particle size distribution (PSD) given in Table 1, with the static AoR of 33°. Meanwhile, experimental data on limestone is provided by Shi et al. [19], specifically the Eskal 150 limestone, since this material has a similar experimental static AoR and density, while the PSD is in a much lower range. The density of Eskal 150 is 2761 kg/m^3, and the static AoR is also 33°.

Material	Diameter (µm)	Cumulative volume distribution (%)
Limestone	97	10
	138	50
	194	90
Quartz sand	300	6.21
	425	24.50
	500	50.55
	600	100

Table 1: Particle Size Distribution of materials.

The experimental data described above will be used as constant input values for each DEM simulation. In addition, four variables will be tested to determine their respective static AoR:

- Restitution coefficient: Ratio between the velocity of the particle before and after collision. The restitution coefficient is in the range of 0 to 1, with 1 denotes a perfectly elastic collision.
- Sliding friction: Force that acts on the opposite direction of the movement between two particles when they collide.
- Rolling friction: Force resists the rolling motion of the particle.
- Bond number: Ratio between gravitational force that act on the particle and the surface tension force, i.e., the easiness of movement due to gravity.

In addition to the range bound of Restitution Coefficient, all other microparameters has a positive bound - the value ranges from 0 to ∞, however in most realistic case the value is also smaller than 1.

4 Calibration of Discrete Particle Model

4.1 GrainLearning

GrainLearning is a calibration toolbox developed by Cheng et al., utilizes the recursive Bayesian algorithm to estimate the uncertainty parameters in DPM. Initially, a wide range of parameter space is quasi-randomly sampled from the initial guess range to create a prior distribution of each parameter. Then, conditioned on the experimental values, the posterior distribution of the parameters is updated recursively by Sequential Monte-Carlo Filtering (SMC Filter) and fitted to a Gaussian Mixture Model. This process is done iteratively, until a desired value that minimises the loss function is reach, typically 3 iterations. Algorithm 1 and the following sections will describe in brief the calibration workflow implemented in GrainLearning.

Algorithm 1 GrainLearning

Input:
y: Experimental values
$\mathbf{x} = \mathbf{F}(\Theta)$: DEM solver
$(\Theta_{min}, \Theta_{max})$: Initial guess range
Main:
▷ *Set uniform prior distribution:* $p(\Theta) = \mathcal{U}(\Theta_{min}, \Theta_{max})$
for k in range $(0, K)$ **do**
▷ *Sampling parameters:*
if $k = 0$ **then** sample N_p parameters values from initial distribution: $\Theta_k^{(i)} \sim p_0(\Theta)$
else if $k > 0$ **then** sample N_p parameters values from prior distribution: $\Theta_k^{(i)} \sim p_{k-1}(\Theta \mid y_{1:T})$
▷ *Evaluate DPM:* $x_k^{(i)} = F(\Theta_k^{(i)})$
▷ Optimizing σ:
while True **do**
▷ *Compute likelihood (Eq. 4):* $p(y_t \mid \Theta_k^{(i)}) \propto \mathcal{N}(y_{kt} \mid x_{kt}^{(i)}, \Sigma)$
▷ *Compute posterior distribution of of* $\Theta_k^{(i)}$ *conditioned to y (Eq. 5).*
▷ *Compute Effective Sample Size (ESS) with Eq. 6.*
▷ *Stop if target ESS value is reached:*
if $k = 0$ and $ESS > 20\%$ **then** break
else if $k > 0$ and $ESS \sim ESS_{max}$ **then** break
▷ *Fit sampled posterior distribution to Gaussian Mixture Model:* $p(\Theta \mid y) = \sum_\alpha^k \lambda_\alpha \mathcal{N}(\mu_\alpha, \sigma_\alpha)$
▷ *Set new prior distribution:* $p(\Theta) \leftarrow p(\Theta \mid y)$
Output: Θ_{opt} in Θ_K that minimizes $|F(\Theta_{opt} - y)|$

4.1.1 Posterior distribution calculation

Initially, the measurement is assumed to have an error represented by a covariance matrix $\Sigma_\alpha = \sigma\omega_\alpha y_\alpha$, with σ the covariance parameter, and important weight of the measurement ω. With Σ, the likelihood of a given state $\Theta_k^{(i)}$, i.e., the probabilistic prediction to the experimental data y can be estimated by the multivariate normal distribution, with y_t measurement data at time step t, and d the dimension of the state vector $\Theta_k^{(i)}$:

$$p(y_t \mid \Theta_k^{(i)}) \propto \frac{1}{(2\pi)^{d/2}|\Sigma|} exp\left(-\frac{1}{2}(y_{kt} - x_{kt}^{(i)})^\intercal \Sigma^{-1}(y_{kt} - x_{kt}^{(i)})\right) \tag{4}$$

With the calibration system being modelled as a hidden Markov model, the posterior distribution of $\Theta_k^{(i)}$ can be calculated using recursive Bayes' rule:

$$p(\Theta_k^{(i)} \mid y) \propto \prod_{t=1}^{N_t} p(y_t \mid \Theta_k^{(i)}) p(\Theta_k^{(i)}) \tag{5}$$

4.1.2 Effective multi-level sampling

The Effective Sample Size (ESS) is calculated by summing the posterior distribution squared of all the sampled parameters value N_p:

$$ESS = \frac{1}{N_p \sum_{i=1}^{N_p} p(\Theta_k^{(i)} \mid y)^2} \tag{6}$$

The main idea is to draw the sample from the previously acquired knowledge about the relationship between Θ and y. In the first iteration ($k = 0$), The uniform prior distribution is chosen as the proposal density, and the parameter spaces are drawn from there. Subsequently, for $k > 0$, the proposal density will be the posterior distribution from the previous iteration $p(\Theta \mid y)$. After each iterations, the sampling space will get narrower - therefore, to ensure a proper proposal density for the sampling of parameters, the optimization process will be continued until appropriate σ which maximizes ESS is reached.

4.1.3 Identification of microparameters with GrainLearning

In the first iteration of calibration, GrainLearning will initialize a set of parameter combinations using Halton sequence, from the initial guess range provided. This set of parameters will be passed to MercuryDPM to analyze with a Heap test, after which a static AoR is produced. From this data, GrainLearning will compute the next set of parameters on the basis of the previous MercuryDPM output, according to algorithm 1. For each attempt, GL will be running for 4 iterations - except when the simulations of that iteration takes more than two days, and that attempt will be classified as failed.

4.2 Neural Network

Artificial Neural Network (ANN) is a set of algorithms that seeks to identify correlations in data utilizing a technique inspired by how the human brain operates - mimicking how each neuron in the brain signals each other. The most basic ANN model is the Feed-forward Multilayer Perceptron Neural Network (MLPNN), in which the purpose is to define the mapping between the input and output $y = f(x; \theta)$ and approximate the parameter θ which results in the best possible function. In MLPNN, the data will flows in one direction from the input to the output, hence the name feed-forward. Like other supervised learning algorithms, an MLPNN needs to be trained before accurately describing the input and output relations. This is typically done by feeding the network with pre-labeled data, comparing the model's output with the desired output, and updating the weights parameter θ - a process called backpropagation. In this assignment's context, Neural Network (NN) will be used when referring to Feedforward Multilayer Perceptron Neural Network, and NN models implemented in this research are provided by the open-source library `TensorFlow` [20].

4.2.1 Designing a Neural Network

There are no general rules for determining the number of layers and the number of neurons per layer, and it depends heavily on each use case. While Benvenuti et al. [7], He et al. [8], and Daniel et al. [9] used only a single-layer ANN and varied the number of neurons, Ye et al. [10] vary both. However, the ultimate goal in both case is to find the combinations which result in the minimum error while also avoiding overfitting, i.e., the model excels on training but perform poorly on the validation step. In this case, for each simulation material, 250 models ranging from 2 to 15 layers and 5 to 15 neurons per layer are tested to determine the best model. Each model is trained for 50 epochs with a batch size of 32, and the metric used to grade the model is Mean Absolute Error. In addition, 20% of the data will be saved for validation of the model.

Another important component of a Neural Network is the activation function. Since each neuron performs calculation by multiplying the input with weight and adding a bias, the activation function's

role would be introducing a non-linearity element into an otherwise linear neuron. According to Goodfellow et al., [21], Rectified Linear Unit (ReLU) is the recommendation for most Deep Learning models, with its ability to preserve much of the properties due to its near-linear shape. ReLU activation function is defined as $f(x) = max(0, x)$.

4.2.2 Training and evaluation method

To train the Neural Network model, 500 DEM simulations with randomized combinations of input parameters have been performed, in order to create a database which maps DEM microparameters to static AoR. Other 125 simulations **did not finish on the time constraint set, and as a results marked as an inaccurate combination.** Afterwards, over 1,000,000 different combinations, as describe in table 2, will be processed by the NN model. Any combinations that producess a static AoR within the 0.1% margin of error to the experimental data is marked as a correct combination. This combination will then be evaluated independently by a DEM simulation, to verify the ability of NN model correctly describes bulk behavior of a material based on the given contact law.

	Restitution Coefficient	Sliding Friction	Rolling Friction	Bond number
Range	[0.5 1]	[1e-5 1]	[1e-5 1]	[1e-5 1]
Number of values	30	30	30	30

Table 2: Random evenly-spaced microparameters combinations

4.3 Random Forest algorithm

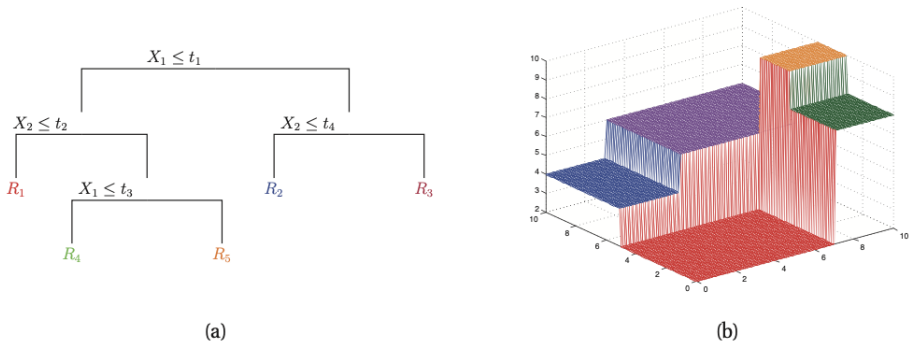

Figure 4: Example of a decision tree regressor on a two-input problem. [22]

This section will discuss different concepts of a Random Forest (RF) algorithm, starting with the basis of the RF: decision tree. Decision tree is an algorithm that generates a tree graph of decisions based on the input provided and their possible outcomes, and as a consequence, it partitions the input space into multiple regions, with each region accounting for a different outcome [22]. An example of a simple decision tree based on two inputs is shown in figure 4.

The most significant advantage of the decision tree, and subsequently, random forest algorithm, is that it is relatively simple, explainable, easy to train and interpolate with little computational resources. However, one crucial drawback of a decision tree is its instability: minor data changes might affect the tree structure, making the decision tree a high variance estimators [22]. Attempts have been made to reduce the uncertainty of the decision tree, one of which is the so-called Random Forest algorithm, which Breiman proposed in 2001 [23]. RF made up for the high variance of a single decision tree by averaging the results over a "forest" of decision trees, with each tree represents an independent sampled vectors.

The concept of decision tree and RF, therefore, fit within the scope of the calibration problem: each DEM physical property of the material is responsible for the static AoR.

4.3.1 Training and evaluation method

Data that is used to train the NN model will also be used to train the RF model. However, as mentioned above, the simple nature of RF allows for faster prediction, therefore over 2.5 million different combinations can be processed with the RF model, and similar to the evaluation method of NN model, the combinations that produces a static AoR within 0.1% will be marked as a correct combination, which will be validate using another full DEM simulation.

5 GrainLearning evaluation

Material	Quartz Sand			Limestone		
Attempt	1	2	3	1	2	3
Restitution Coefficient	[0.5 1]	[0 1]	[0 1]	[0.5 1]	[0.5 1]	[0 1]
Rolling Friction	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]
Sliding Friction	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]
Bond number	[0 1]	[0 0.5]	[0.5 1]	[0 1]	[0 0.5]	[0.5 1]

Table 3: Calibration attempts using GL

Table 3 gives details on different calibrations with GL for two materials mentioned above: quartz sand and limestone. The difference between each attempt is that the search range of the microparameters changed. The first attempt serves as the control for both materials, while the second and the third are set up in smaller ranges. Overall, approximately 200 DEM simulations were performed for quartz sand, and 3 produced a valid combination. For limestone, 300 DEM simulations were performed, and five are valid. The valid combinations are marked in bold on the result tables in the following section.

5.1 Limestone

The calibration results by GL for limestone are described in table 4, and the details on how the sampling algorithm performs, i.e., conditioned on the previous simulation, is the sampled parameters for the next iteration make the simulation result converge to the experimental result, are described in figure 5. In the third calibration attempt, the second iteration did not finish in time due to the system's high level of kinetic energy; therefore, only iteration 1 is shown. In the control attempt, initially, only four iterations were performed. However, one remarkable observation is that GL clusters over the combinations produce a static AoR around 40°. This is reflected in the second iteration's result, where the best combination results in a static AoR of 39.4803°. Moreover, although the third iteration's result is 33.0979°, this seems like an outlier of the cluster. Therefore, an additional iteration was performed - and the results here verify the observation. The closest value to experimental static AoR is 31.5243°, and it is also the outlier of the cluster.

After the first calibration attempt, it is clear that the combinations that would result in the desired static AoR lie around $0.8 - 1.0$ for restitution coefficient and in the lower-0.25 range for the rest of the contact parameters. Therefore, two more attempts were performed, with different initial ranges: attempt 2 with sliding friction, rolling friction, and bond number ranging from 0 to 0.5. With attempt 3, the sliding friction, rolling friction, and bond number range from 0.5 to 1, while the restitution coefficient's range is widened to 0 to 1.

As expected, the second attempt's performance was the most robust, reaching a near-perfect solution at the end of iteration 4. The clustering of the optimal result can also be seen clearly in figure 5. Meanwhile, the results from the third attempt verify the conclusion from the first attempt about the

range of the optimal parameters - with higher rolling friction, sliding friction, and bond number, the static AoR reaches a maximum value of around 65°.

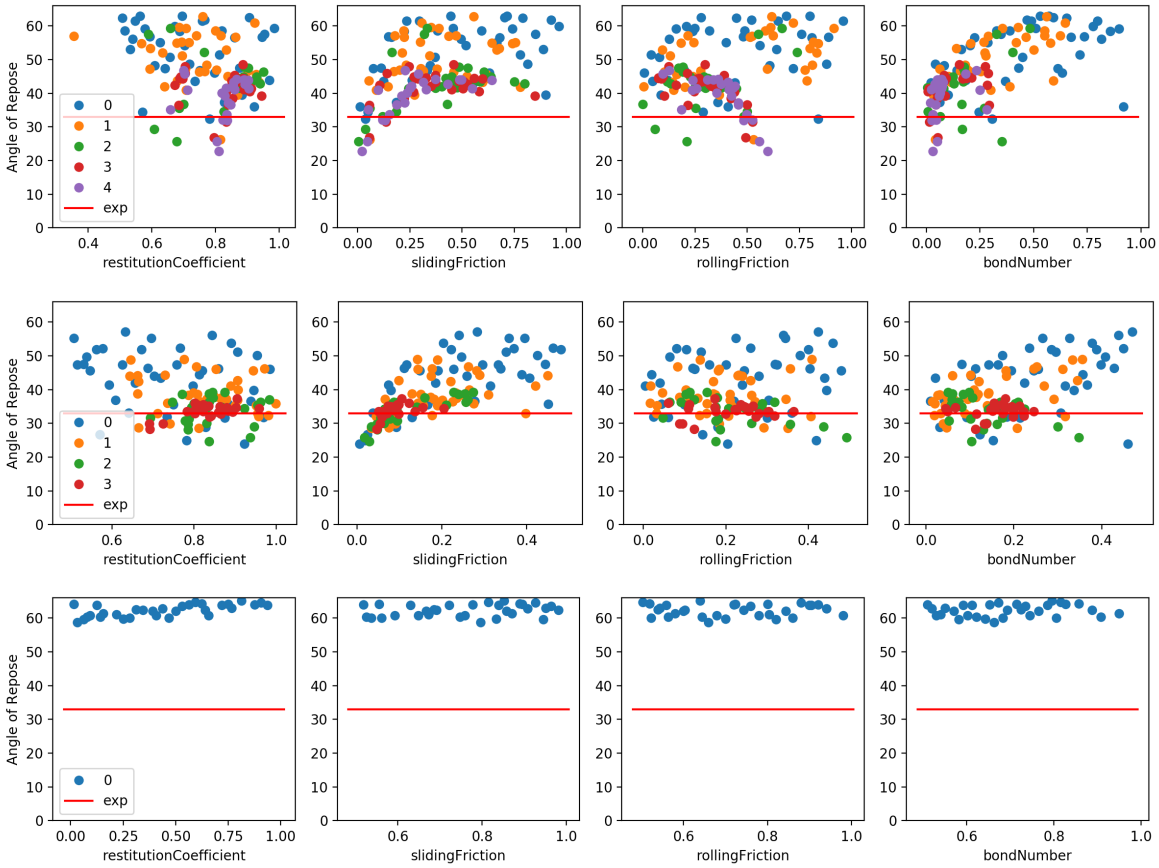

Figure 5: *From top to bottom,* attempt 1, 2, and 3 at calibrating Eskal with GL. Scatter dots with different colors denote each iteration, and red line denotes experimental value.

Attempt/Iter	Restitution Coefficient	Sliding Friction	Rolling Friction	Bond number	Result AoR
1.1	0.7812	0.037	0.84	0.3061	32.3163
1.2	0.869	0.2725	0.3652	0.0325	39.4803
1.3	0.831	0.1194	0.484	0.064	33.0406
1.4	0.8332	0.135	0.5262	0.0137	31.5243
2.1	0.6406	0.037	0.36	0.3061	33.0979
2.2	0.9546	0.3983	0.032	0.0334	32.9993
2.3	0.8263	0.0917	0.2226	0.1411	34.1421
2.4	0.8025	0.0710	0.2802	0.1748	32.9812
3.1	0.0312	0.7963	0.66	0.6632	58.6606

Table 4: Calibration results of limestone with GL.

5.2 Quartz sand

The calibration result for quartz sand is given in table 5, and the parameters sampling graph is given in figure 6. In the control attempt, only the first iteration is shown, partly due to 6/40 simulations of

the second iteration does not finish in time, but also due to the results of the second iteration does not cluster at the experimental value, with most averaging around 45° to 50°. Due to the similarity between quartz sand and limestone in terms of experimental static AoR, the second attempt of quartz sand will be initialized with the same range as the second attempt of limestone. And as a result, attempt 2 has the best performance out of the three. One noticeable thing here is that the rolling friction has two different clusters identified by GL instead of one, compared to sliding friction, restitution coefficient, or bond number. This denotes the multi-solution phenomena of a calibration problem since there could be more than one combination that can produce a sufficient static AoR - which is shown in iterations 3 and 4 of attempt 2: the rolling friction for iteration 3 is 0.07, while for iteration 4 is 0.41. Different experiments, i.e., Shear Cell test or Drum test (Dynamic AoR), would be needed in addition to the heap test to find an ideal combination of microparameters. However, this is out of the scope of the current research.

In the third attempt, although the range was specified as shown in table 3, the Gaussian Mixture Model algorithm of GL sampled some of the values in iteration 3 and 4 outside the initial range. This was a known bug in the current GL version implemented in MercuryDPM. However, it was able to generate a correct combination which results in a static AoR of 34.2227°, remarkably close to the experimental value, by sampling a combination with very low sliding friction.

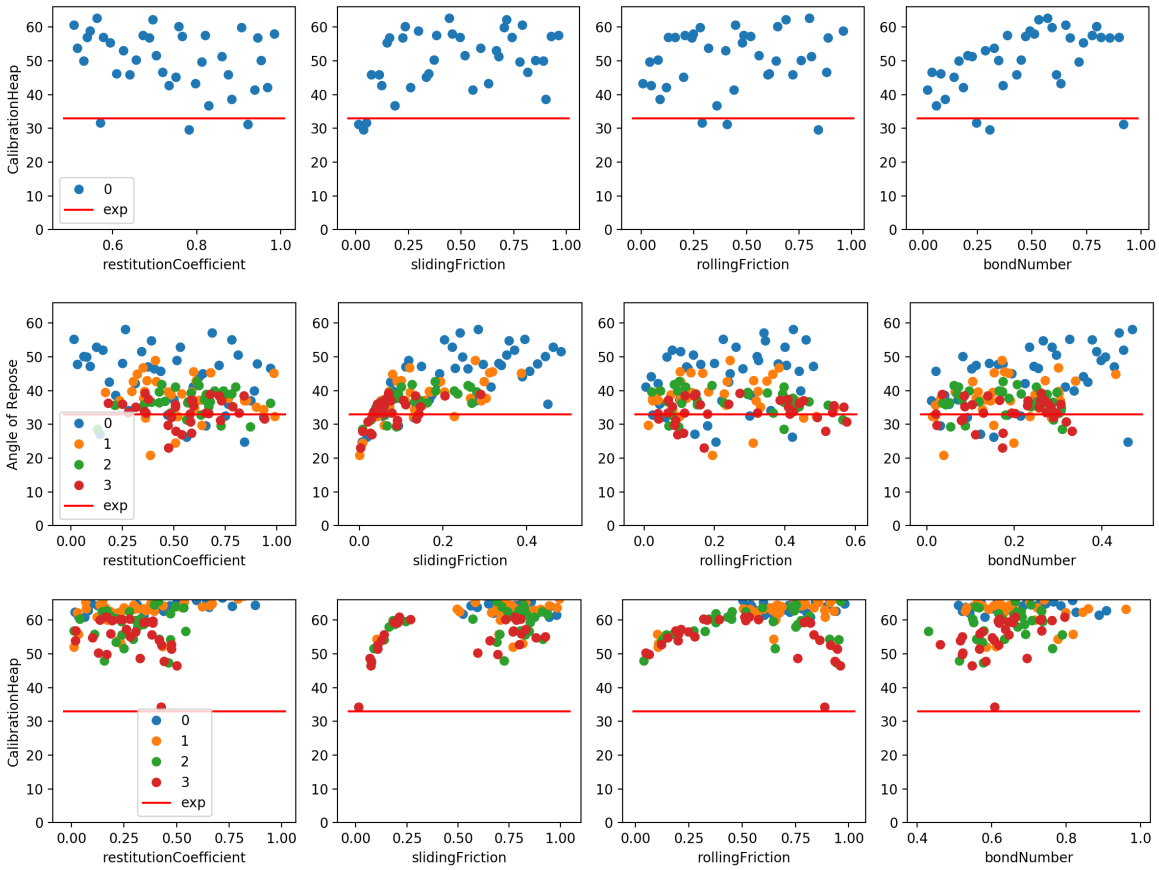

Figure 6: *From top to bottom,* attempt 1, 2, and 3 at calibrating quartz sand with GL. Scatter dots with different colors denote each iteration, and red line denotes experimental value.

5.3 GL discussion

Throughout multiple calibration routines with GrainLearning, it is identified that the initial guessing range played a key role in the ability of GL to idenfity the correct set of micro-parameters.

Attempt/Iter.	Restitution Coefficient	Sliding Friction	Rolling Friction	Bond number	Result AoR
1.1	0.5703	0.0493	0.288	0.2448	31.6303
2.1	0.4688	0.0617	0.024	0.1836	32.8394
2.2	0.992	0.2261	0.0987	0.0126	32.3687
2.3	0.6307	0.0606	0.0787	0.1795	32.8670
2.4	0.4800	0.0321	0.413	0.2946	33.0521
3.1	0.0625	0.9444	0.82	0.5816	60.9714
3.2	0.0109	0.7683	0.1052	0.5853	52.0551
3.3	0.4604	0.0752	0.9406	0.5777	47.3408
3.4	0.427	0.0146	0.8872	0.6075	34.2227

Table 5: Calibration results of sand with GL.

6 Supervised model evaluation

In this section, the performance of the two supervised models, i.e., Neural Network and Random Forest, is investigated for limestone and quartz sand, respectively.

6.1 Limestone

For limestone, 12 over 500,000 combinations of DEM input parameters processed by the ANN was a 'valid' combination: the output of the ANN was $33 \pm 0.01°$. Meanwhile, with 2,500,000 million combinations processed by RF, 22 of them were valid combinations - however, many of them are closely similar with a minor difference in one of the micro parameters, and only nine are distinct. The valid combinations are described in table 6 and 7, with figure 7 illustrates the combinations and their respective output. Overall, the NN model has correctly identified three combinations, while the Random Forest model has 2.

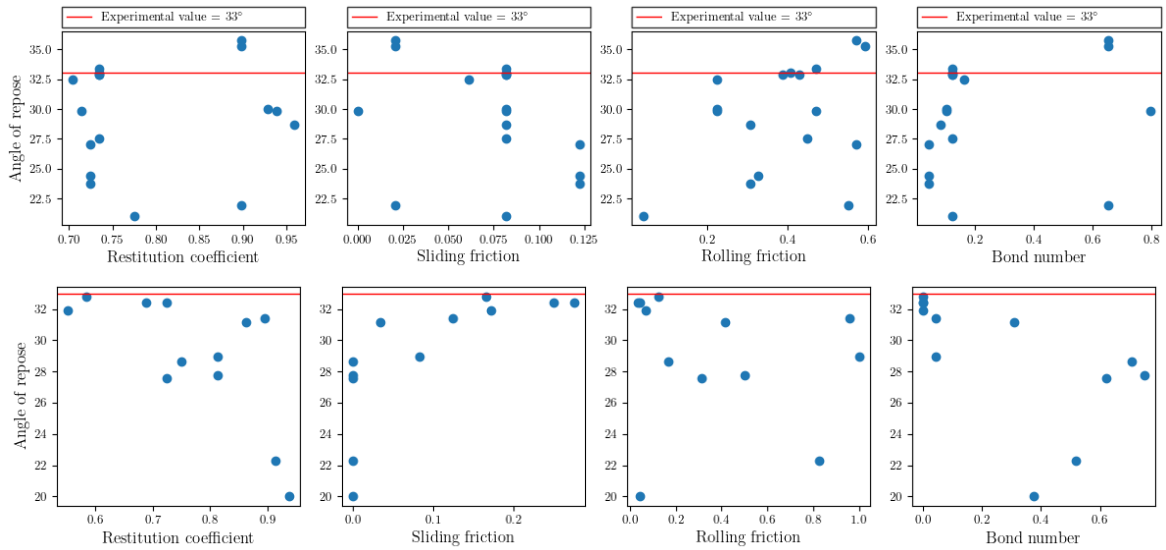

Figure 7: *From top to bottom,* valid contact law parameters identified by Random Forest and Neural Network model for limestone, and their respective simulation results.

Restitution coefficient	Sliding friction	Rolling friction	Bond number	Angle of repose
0.6875	**0.2500**	**0.0417**	**0**	**32.4263**
0.7500	0	0.1667	0.7083	28.6156
0.5833	**0.1667**	**0.1250**	**0**	**32.7926**
0.5517	0.1724	0.0690	0	31.9425
0.8125	0.0833	1.0000	0.0417	28.9393
0.7241	**0.2759**	**0.0345**	**0**	**32.3962**
0.8958	0.1250	0.9583	0.0417	31.4044
0.8621	0.0345	0.4138	0.3104	31.1331
0.7241	0	0.3104	0.6207	27.5978
0.8125	0	0.5000	0.7500	27.7676
0.9138	0	0.8276	0.5172	22.2677
0.9375	0	0.0417	0.3750	20.0361

Table 6: Valid contact law parameters identified by the NN model for limestone and their respective simulation results.

Restitution coefficient	Sliding friction	Rolling friction	Bond number	Angle of repose
0.7041	**0.0612**	**0.2245**	**0.1633**	**32.4872**
0.7245	0.1225	0.5714	0.0408	27.0585
0.7347	**0.0816**	**0.4082**	**0.1225**	**33.0331**
0.7347	0.0816	0.4694	0.1225	33.3593
0.7755	0.0816	0.0408	0.1225	21.0172
0.8980	0.0204	0.5714	0.6531	35.7342
0.9286	0.0816	0.2245	0.1020	29.9625
0.9592	0.0816	0.3061	0.0816	28.6865
0.7143	0	0.4694	0.7959	29.8154

Table 7: Valid contact law parameters identified by the RF model for limestone and their respective simulation results.

6.2 Quartz Sand

It is noteworthy that the quartz sand model was trained with less simulations compare to limestone model, with 322 DEM simulations. However, the performance of NN model when predicting the correct combinations that results in a static AoR of 33°: 4 out of 10 combinations are valid after verified by a full DEM simulation. Meanwhile, while the RF model predicts 20 different combinations, only 3 of them were valid - but interestingly, most of the combinations predict by RF model ranging very close to the experimental value, from 29° to 31°. The number of combinations passed to NN and RF model for quartz sand is approximately the same as for limestone.

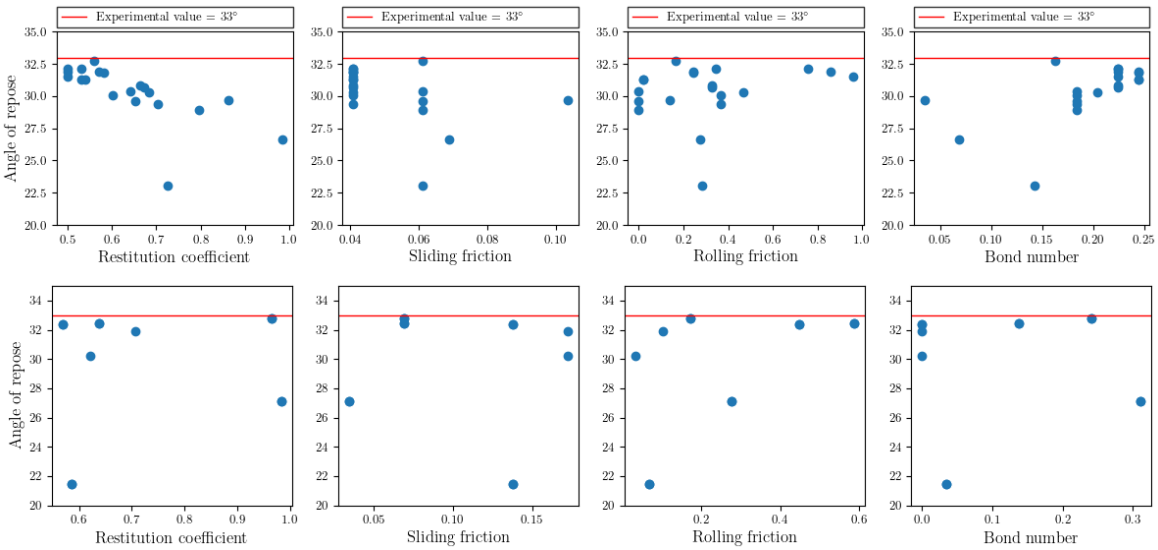

Figure 8: *From top to bottom,* valid contact law parameters identified by Random Forest and Neural Network model for quartz sand, and their respective simulation results.

Restitution coefficient	Sliding friction	Rolling friction	Bond number	Angle of repose
0.5690	**0.1379**	**0.4483**	**0**	**32.3822**
0.7069	0.1724	0.1035	0	31.9092
0.6207	0.1724	0.0345	0	30.2535
0.6379	**0.0690**	**0.5862**	**0.1379**	**32.4170**
0.5862	0.1379	0.0690	0.0345	21.4586
0.9655	**0.0690**	**0.1724**	**0.2414**	**32.7969**
0.9828	0.0345	0.2759	0.3104	27.0953
0.5862	0.1379	0.0690	0.0345	21.4586
0.9655	**0.0690**	**0.1724**	**0.2414**	**32.7969**
0.9828	0.0345	0.2759	0.3104	27.0953

Table 8: Valid contact law parameters identified by the NN model for quartz sand and their respective simulation results.

Restitution coefficient	Sliding friction	Rolling friction	Bond number	Angle of repose
0.5612	**0.0612**	**0.1633**	**0.1633**	**32.7325**
0.5000	**0.0408**	**0.7551**	**0.2245**	**32.1187**
0.5000	0.0408	0.8571	0.2245	31.8814
0.5306	**0.0408**	**0.3469**	**0.2245**	**32.1125**
0.6429	0.0612	0	0.1837	30.3771
0.5714	0.0408	0.2449	0.2449	31.8564
0.6020	0.0408	0.3674	0.1837	30.0461
0.5408	0.0408	0.0204	0.2449	31.2764
0.5000	0.0408	0.9592	0.2245	31.5459
0.5816	0.0408	0.2449	0.2449	31.8087
0.6531	0.0612	0	0.1837	29.6271
0.7041	0.0408	0.3674	0.1837	29.3786
0.6633	0.0408	0.3265	0.2245	30.8579
0.5306	0.0408	0.0204	0.2449	31.2787
0.6735	0.0408	0.3265	0.2245	30.6531
0.7959	0.0612	0	0.1837	28.9042
0.6837	0.0408	0.4694	0.2041	30.2650
0.7245	0.0612	0.2857	0.1429	23.0546
0.8621	0.1035	0.1379	0.0345	29.6629
0.9828	0.0690	0.2759	0.0690	26.6594

Table 9: Valid contact law parameters identified by the RF model for quartz sand and their respective simulation results

6.3 NN and RF model discussion

Overall, the supervised models have demonstrated the ability to learn the contact law relationship of DPM for a specified materials. It is expected that the model would not has a high predition accuracy ($> 70\%$) when being fed and process millions of combinations, with limited training data. And therefore, limitations of the model can be seen in some case, especially with quartz sand: the model

7 Discussion

- GrainLearning has an inconsistent performance, especially when parameter ranges are specified too wide. Assume a calibration with GL cost 40 DEM simulations per iteration, for each bulk parameter it would be 7-8 iterations 300 DEM simulations = ¿ less DEM simulations, fully automatic, has been proven for more complex contact laws.

- For limestone, the amount of DEM simulations used to train NN and RF are 485, and for quartz sand are 322. The numbers are slightly higher than GL, but the performance are more robust from a small experiment: Produces multiple valid combinations per bulk parameter, therefore easier to identify the

correct one by comparision with other bulk parameter's combinations.

- Limitations of RF: has not been tested against complex contact laws

8 Conclusion

References

[1] A. B. Yu, F. Bassani, G. L. Liedl, P. Wyder, Powder Processing: Models and Simulations, Encyclopedia of Condensed Matter Physics, Elsevier, Oxford, 2005, pp. 401–414. doi:10.1016/B0-12-369401-9/00556-8.

[2] E. S. Oran, J. P. Boris, R. A. Meyers, Encyclopedia of Physical Science and Technology (Third Edition), Academic Press, New York, 2002, Ch. Fluid Dynamics, pp. 31–43. doi:10.1016/B0-12-227410-5/00248-9.

[3] R. M. Nedderman, Statics and Kinematics of Granular Materials, Cambridge University Press, Cambridge, 1992, Ch. Introduction, pp. 1–6. doi:10.1017/CBO9780511600043.002.

[4] A. Fernandes, H. Gomes, E. M. B. Campello, P. Pimenta, A fluid-particle interaction method for the simulation of particle-laden fluid problems, 2017. doi:10.20906/CPS/CILAMCE2017-0139.

[5] X. Weng, Modeling of complex hydraulic fractures in naturally fractured formation, Journal of Unconventional Oil and Gas Resources 9 (2015) 114–135. doi:10.1016/j.juogr.2014.07.001.

[6] P. A. Cundall, O. D. L. Strack, A discrete numerical model for granular assemblies, Géotechnique 29 (1) (1979) 47–65. doi:10.1680/geot.1979.29.1.47.

[7] L. Benvenuti, C. Kloss, S. Pirker, Identification of dem simulation parameters by artificial neural networks and bulk experiments, Powder Technology 291 (2016) 456–465. doi:10.1016/j.powtec.2016.01.003.

[8] P. He, Y. Fan, B. Pan, Y. Zhu, J. Liu, D. Zhu, Calibration and verification of dynamic particle flow parameters by the back-propagation neural network based on the genetic algorithm: Recycled polyurethane powder, Materials 12 (20) (2019). doi:10.3390/ma12203350.

[9] D. Schiochet Nasato, R. Queiroz Albuquerque, H. Briesen, Predicting the behavior of granules of complex shapes using coarse-grained particles and artificial neural networks, Powder Technology 383 (2021) 328–335. doi:10.1016/j.powtec.2021.01.029.

[10] F. Ye, C. Wheeler, B. Chen, J. Hu, K. Chen, W. Chen, Calibration and verification of dem parameters for dynamic particle flow conditions using a backpropagation neural network, Advanced Powder Technology 30 (2) (2019) 292–301. doi:10.1016/j.apt.2018.11.005.

[11] H. Q. Do, A. M. Aragón, D. L. Schott, A calibration framework for discrete element model parameters using genetic algorithms, Advanced Powder Technology 29 (6) (2018) 1393–1403. doi:10.1016/j.apt.2018.03.001.

[12] H. Cheng, T. Shuku, K. Thoeni, H. Yamamoto, Probabilistic calibration of discrete element simulations using the sequential quasi-monte carlo filter, Granular Matter 20 (1) (2018) 11. doi:10.1007/s10035-017-0781-y.

[13] T. F. Teferra, Chapter 3 - engineering properties of food materials, in: M. Kutz (Ed.), Handbook of Farm, Dairy and Food Machinery Engineering (Third Edition), third edition Edition, Academic Press, 2019, pp. 45–89. doi:doi.org/10.1016/B978-0-12-814803-7.00003-8.

[14] M. Rackl, F. E. Grötsch, 3d scans, angles of repose and bulk densities of 108 bulk material heaps, Scientific Data 5 (1) (2018) 180102. doi:10.1038/sdata.2018.102.

[15] T. Weinhart, L. Orefice, M. Post, M. P. van Schrojenstein Lantman, I. F. Denissen, D. R. Tunuguntla, J. Tsang, H. Cheng, M. Y. Shaheen, H. Shi, P. Rapino, E. Grannonio, N. Losacco, J. Barbosa, L. Jing, J. E. Alvarez Naranjo, S. Roy, W. K. den Otter, A. R. Thornton, Fast, flexible particle simulations - An introduction to MercuryDPM, Computer Physics Communications 249 (2020) 107129. doi:10.1016/j.cpc.2019.107129.

[16] S. Luding, Cohesive, frictional powders: contact models for tension, Granular Matter 10 (4) (2008) 235. doi:10.1007/s10035-008-0099-x.

[17] H. A. Navarro, M. P. de Souza Braun, Determination of the normal spring stiffness coefficient in the linear spring–dashpot contact model of discrete element method, Powder Technology 246 (2013) 707–722. doi:10.1016/j.powtec.2013.05.049.

[18] S. M. Derakhshani, D. L. Schott, G. Lodewijks, Micro–macro properties of quartz sand: Experimental investigation and dem simulation, Powder Technology 269 (2015) 127–138. doi:10.1016/j.powtec.2014.08.072.

[19] H. Shi, G. Lumay, S. Luding, Stretching the limits of dynamic and quasi-static flow testing on cohesive limestone powders, Powder Technology 367 (2020) 183–191. doi:10.1016/j.powtec.2020.03.036.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015). doi:10.5281/zenodo.4724125. URL https://www.tensorflow.org/

[21] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[22] K. Murphy, Machine Learning: A Probabilistic Perspective, Adaptive Computation and Machine Learning series, MIT Press, 2012.

[23] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.